GRIB floating point compression

- GRIB uses lossy compression of floating point data
  - Store integers; \( f = i \times \text{scale} + \text{offset} \)
  - Bounds the absolute precision: \( \text{abs}(f_{\text{org}} - f) \leq \text{scale}/2 \)
- GRIB-1 uses bit-packing
- GRIB-2 uses JPEG-2000 wavelet compression
- GRIB has excellent compression
  - On our test NCEP data, GRIB is 2.5-25x smaller than uncompressed single precision floating point, eg netCDF-3
  - Recent NCEP model runs (15 Grib-1, 46 Grib-2, 26 Gbytes)
- Can netCDF-4 get close to this?
  - JPEG-2000 considered patent encumbered (?)
  - What about other compression?
average = 8.9
stdev = 5.5
Current netCDF-4 (deflate) ratio netCDF4 / GRIB

Weighted average ratio = 1.32
Total # grib1 records = 24,933

Weighted average ratio = 2.24
Total # grib2 records = 375,470
Other possibilites

Other compression algorithms
• bzip2
• LZMA (7zip)

Lossy compression techniques
• bit shaving (set low order bits to 0)
• scale/offset (turn floats into ints)

Testing methodology
• all in Java : expect to be good estimate of C library
• read GRIB, use Java compression libraries
  • floats as they are returned from GRIB reader (limited precision)
  • floatShaved: use Nbits from GRIB, set lower bits to 0
  • ints: use exact same integer array as GRIB
File size ratio with GRIB2 JPEG2k
On limited precision floats
(Java)
Total file sizes ratio with GRIB2 JPEG2k
(Java)

- **floats**
- **floatShaved**
- **scale/offset ints**

Legend:
- **deflate**
- **bzip2**
- **Izma**
<table>
<thead>
<tr>
<th></th>
<th>size (GB)</th>
<th>uncompressed</th>
<th>compress</th>
</tr>
</thead>
<tbody>
<tr>
<td>deflate floats</td>
<td>35.90</td>
<td>2.28</td>
<td>14.71</td>
</tr>
<tr>
<td>deflate floatShaved</td>
<td>34.38</td>
<td>1.98</td>
<td>13.59</td>
</tr>
<tr>
<td>deflate ints</td>
<td>33.98</td>
<td>1.89</td>
<td>11.96</td>
</tr>
<tr>
<td>bzip2 floats</td>
<td>19.50</td>
<td>17.80</td>
<td>55.84</td>
</tr>
<tr>
<td>bzip2 floatShaved</td>
<td>18.18</td>
<td>15.20</td>
<td>48.86</td>
</tr>
<tr>
<td>bzip2 ints</td>
<td>18.32</td>
<td>14.17</td>
<td>43.09</td>
</tr>
<tr>
<td>lzma floats</td>
<td>22.40</td>
<td>14.50</td>
<td>473.19</td>
</tr>
<tr>
<td>lzma floatShaved</td>
<td>20.64</td>
<td>12.31</td>
<td>454.08</td>
</tr>
<tr>
<td>lzma ints*</td>
<td>19.05</td>
<td>12.94</td>
<td>482.02</td>
</tr>
<tr>
<td>grib</td>
<td>17.12</td>
<td>23.53</td>
<td></td>
</tr>
</tbody>
</table>
File size ratio NetCDF-4 / GRIB
bzip2 on floats

<table>
<thead>
<tr>
<th></th>
<th>avg</th>
<th>stddev</th>
</tr>
</thead>
<tbody>
<tr>
<td>total</td>
<td>1.12</td>
<td>0.36</td>
</tr>
<tr>
<td>grib1</td>
<td>0.92</td>
<td>0.27</td>
</tr>
<tr>
<td>grib2</td>
<td>1.20</td>
<td>0.37</td>
</tr>
</tbody>
</table>
Conclusions

➔ On NCEP Model GRIB files “limited precision” floats
  ◆ Bzip2 can get to within 20% of GRIB on average
  ◆ Ratios of bzip2/grib vary between .4 and 1.7
➔ Bzip2 looks like a good candidate to add as a standard compression option in netCDF-4
  ◆ tradeoff files size and un/compress times
➔ We are considering a “lossy compression” option in netCDF-4 using bit shaving and/or scale/offset
  ◆ expect bzip2 within 10% of GRIB-2 JPEG-2000
➔ Possible utility to copy GRIB to netCDF-4 and get the exact floating point numbers back
➔ Other compression options still to explore
  ◆ fpzip, zfp from Peter Lindstrom at LLNL
  ◆ ??